Protection of LAN-wide, P2P interactions: a holistic approach

André Zúquete

IEETA/IT,
University of Aveiro,
Campus Universitário de Santiago,
3810-193 Aveiro, Portugal
E-mail: andre.zuquete@ua.pt

Abstract: This article advocates the need of a holistic approach to protect LAN interactions and presents a solution for implementing it based on secure LAN (SLAN), a novel security architecture. SLAN uses the 802.1X access control mechanisms and is supported by a key distribution centre (KDC) built upon an 802.1X authentication server. The KDC is used, together with a new host identification policy and modified DHCP servers, to provide proper resource allocation and message authentication in DHCP transactions. The KDC is used to authenticate ARP transactions and to distribute session keys to pairs of LAN hosts, allowing them to set up arbitrary, LAN-wide peer-to-peer security associations using such session keys. We show how PPPoE and IPSec security associations may be instantiated and present a prototype implementation for IPSec.

Keywords: holistic LAN security; 802.1X framework; SLAN architecture; secure communication networks; P2P security associations.

Reference to this paper should be made as follows: Zúquete, A. (2009) 'Protection of LAN-wide, P2P interactions: a holistic approach', *Int. J. Communication Networks and Distributed Systems*, Vol. 3, No. 4, pp.408–426.

Biographical notes: André Zúquete received his PhD in Informatics and Computer Engineering at the IST, Technical University of Lisbon. He is an Assistant Professor at the Department of Electronics, Telecommunications and Informatics of University of Aveiro. His research interests include security in distributed systems. He has worked on several national and European projects concerned with security issues in distributed environments. He is author of many research studies published in national and international journals, conference proceedings and a book on network security (in Portuguese).

1 Introduction

The internet is usually regarded as an insecure network; on the contrary, LANs (local area networks), when observed in isolation, tend to be considered secure. However, in our opinion, this is a wrong and risky assumption. Even if one knows and trusts the persons sharing the same LAN, computers in a LAN can still be compromised by cyberplagues; therefore, one cannot simply trust only on people for building up the

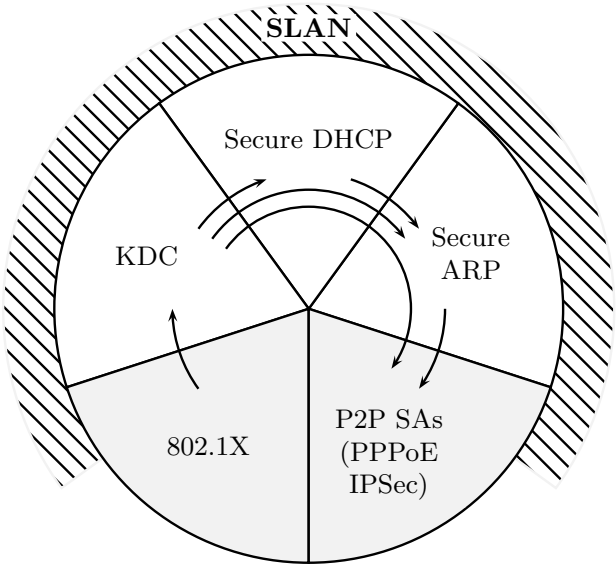
confidence in the correct exploitation of a LAN. Moreover, LANs may be shared by persons that know nothing of each other. For instance, cabled or wireless LANs used in public, shared workspaces, are explored by communities of users that only have in common an authorisation for accessing the network. In this case, LANs cannot be regarded at all as secure network environments by the people using them.

Because LANs are not usually considered as dangerous computing environments, all management and configuration activities occurring between hosts in a LAN are not protected at all. The result is that the management of a LAN is highly sensitive to several local attacks using eavesdropping and origin spoofing techniques.

Spoofing attacks launched locally in a LAN usually require a direct connection of the attacker to the LAN. In other words, it requires an inside attacker. Inside attackers can be prevented to a certain level by physical barriers, ensuring that only authorised persons are connected to the same LAN. However, again, computers in a LAN can be compromised by cyberplagues and remotely managed to act as inside attackers or users sharing the same LAN may not trust or even know each other.

In this paper, we describe a holistic approach for protecting LAN-wide interactions. This approach is relevant and suitable for both cabled and wireless LANs, providing a flexible set of security attributes for protecting LAN-wide interactions. Namely, we show how we may achieve an integrated LAN security, starting from an 801.X authentication for network access, encompassing secure variants of DHCP and ARP configuration protocols, and ending in the automatic deployment of peer-to-peer (P2P), LAN-wide security associations (see Figure 1).

Figure 1 Integration of P2P security association (SAs), PPPoE or IPSec, with the SLAN architecture



Notes: As we can see, SLAN can transparently bridge the gap between user-LAN 802.1X authentication and the deployment and exploitation of LAN-wide, P2P SAs. The grey-filled areas represent existing standards and the white areas represent components of the proposed SLAN architecture. The arrows represent services provided by components/protocols to others.

This integrated security is based on the services provided by the SLAN architecture and on different solutions for implementing security associations (SAs). The rational and the core services of SLAN were first presented by Zúquete and Marques (2006). Here we will go a bit further and, besides providing a more detailed comparison with other related works, we show how secure PPPoE links or IKE/IPSec SAs (Mamakos et al., 1999; Harkins and Carrel, 1998; Thayer et al. 1998) can be deployed and managed using the SLAN key distribution mechanism. Note that we do not present a new standard for implementing SAs; instead, we show how we can integrate the SLAN architecture to configure existing protocols implementing SAs. To facilitate the task of the reader, this article provides a brief presentation of the SLAN architecture.

As a proof of concept, in this article we also describe a prototype, implemented on Linux systems, for testing the automatic instantiation of IPSec SAs between pairs of SLAN hosts using session keys distributed along secure ARP interactions.

The article is structured as follows. Section 2 overviews security threats in cabled and wireless LANs. Section 3 overviews the SLAN architecture. Section 4 describes several different approaches for setting up LAN-wide SAs using SLAN. Section 5 describes a prototype implementation. Section 6 describes related work. Finally, Section 7 presents the conclusions.

2 Security threats in cabled and wireless LANs

Because LANs are not usually considered a dangerous computing environment, traffic between hosts in a LAN is not protected at all. The result is that information traversing the LAN is highly sensitive to several local attacks using eavesdropping, modification and origin spoofing techniques. In this section, we will provide a few examples about what misbehaving LAN hosts can do to their local neighbours.

Data link layer insecurity

Security issues at the data link layer are mainly related with addressing and switching.

Regarding addressing, ARP (Plummer, 1982) is the only protocol responsible for providing address mappings between network and data link layers. Since no message authentication is provided, ARP is vulnerable to spoofing, modifying and replaying attacks. Through the manipulation of ARP request and reply packets, it is possible to corrupt ARP cache tables of remote devices (ARP cache poisoning).

Regarding switching, using switches instead of hubs ensures that frames are only forwarded to the required network segments, which reduces the load of each segment and avoids eavesdropping. However, switches can be mislead by attackers sending bogus frames with spoofed data link (L2) addresses, allowing them to get frames targeted to the spoofed host.

Another way of eavesdropping on a switched network is by 'converting' a switch on a hub by flooding its L2 address table with bogus L2 addresses. In environments with multiple switches, eavesdropping or DoS attacks can also be launched by using spanning-tree protocol manipulation (Dubrawsky, 2004).

Both addressing and switching vulnerabilities can be minimised with virtual LANs (VLANs). VLANs permit a physical network to be divided into a set of smaller logical

networks, making communication between LANs only possible with a router. Because the attack techniques described before are only possible in the data link layer, by using VLANs network administrators reduce the scope of attacks. VLANs, however, suffer the 'VLAN hopping' security issue (Dubrawsky, 2004).

Network layer insecurity

Routing is one major security issue of the network layer. Attackers can configure default gateways of local LAN hosts by deploying rogue DHCP servers (Droms, 1997) or by using false ICMP redirect packets (Postel, 1981), spoofing both IP and L2 addresses of the real gateway. Hosts can be configured to reject all ICMP redirect packets, but that reduces the flexibility in the management of LAN routes.

Rogue DHCP servers can also provide other wrong configuration information, such as fake IP addresses of several network services (e.g., local DNS server).

IPSec (Thayer et al., 1998) is the most accepted solution to mitigate network layer vulnerabilities. IPSec can provide authentication, integrity, confidentiality and anti-replay services, both for IPv4 and IPv6, in two different modes: transport and tunnel. However, IPSec is not easy to manage and is not suitable for protecting DHCP protocol runs (because hosts need to get an IP address prior to use IKE (Harkins and Carrel, 1998), the protocol that negotiates IPSec SAs).

3 SLAN architecture

The core services provided by the SLAN security architecture are the following (Zúquete and Marques, 2006):

1 Authenticated key distribution: distribution of secret, shared keys for pairs of LAN hosts.

2 Secure DHCP: authentication of all replies provided by a genuine DHCP server and detection of replays of past replies.

3 Secure ARP: authentication for ARP request and replies, namely the authentication of the IP-L2 mappings provided and the detection of past requests/replies.

Additionally, SLAN potentiates the protection of other network protocols between any pair of hosts in the LAN. In this case, LAN can provide peers with secret material suitable for enforcing arbitrary LAN-wide SAs. For instance, protected ARP protocol runs distribute such keys.

All these services are supported by secret, shared session keys between interacting hosts. Session keys are associated with pairs of SLAN hosts and distributed by a local SLAN KDC. Hosts authenticate themselves against the KDC using a shared, short-term secret key. Session keys are distributed among pairs of SLAN hosts within DHCP and ARP protocol's messages.

To avoid confusions, hereafter we will use MAC as an acronym of message authentication code and L2 address instead of MAC (medium access control) address. We will also use the expression MAC(K) to denote a MAC computed with key K over the

message the MAC belongs to; and the expression [*D*] + MAC(*K*)to denote the set formed by the datum *D* and its MAC computed with key *K*.

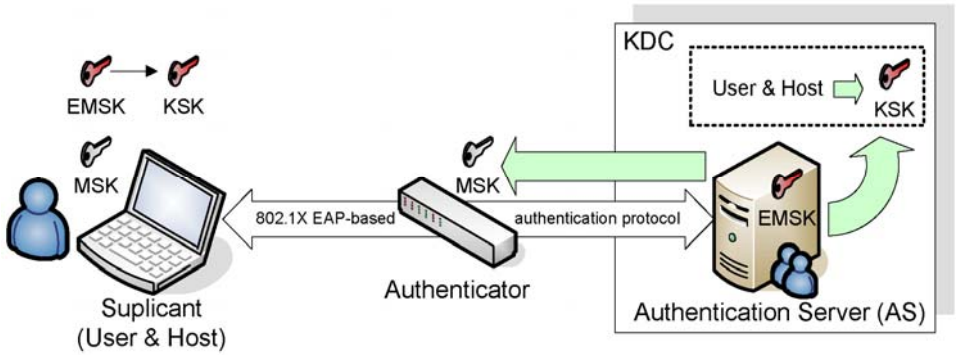
3.1 Key distribution centre (KDC)

The KDC must share a secret with each SLAN-enabled local host. In the SLAN architecture, the KDC is build upon an 802.1X authentication server (AS) (see Figure 2). The SLAN client first runs an EAP-based, 802.1X authentication protocol (Aboba et al., 2004, IEEE, 2001) with the AS/KDC and both end up with a EMSK (extended master session key) after a successful authentication of the former. The EMSK can then be used to derive an authentication secret, KDC session key (KSK), shared between the KDC and an authenticated SLAN host. The KDC associates each KSK to some identification of the user that ran the 802.1X authentication protocol.

Zúquete and Marques (2006) used the pairwise master key (PMK), instead of a key derived from EMSK, for the secret shared between a SLAN client and the KDC. Though both may be used, it is more correct to use a key derived from EMSK, since PMK keys are derived from master session key (MSK) keys transmitted by the AS to 802.1X authenticators (e.g., wireless access points, see Figure 2), so they are not exclusively known by the SLAN client and the KDC. The EMSK, on the contrary, remains exclusively known by the SLAN client and the AS; since the KDC is an extension of the AS, it naturally becomes aware of EMSK keys (or keys derived from EMSK).

SLAN administrators are free to choose the best EAP-based authentication protocols for 802.1X authentication. However, the SLAN trust model requires confidence on the KDC, therefore all EAP-based authentication protocols ran within the 802.1X must require mutual authentication, for properly authenticating the AS/KDC.

Figure 2 Relationship between the 802.1X authentication framework and the SLAN KDC (see online version for colours)



Notes: The KDC is build upon the AS and stores mappings between KSK keys, derived from EMSK keys and the corresponding users and hosts.

3.2 Network identification

Since L2 addresses can be manipulated at will, it is possible to have two hosts with equal L2 addresses in the same LAN without being able to decide which one is legitimate, if any. Therefore, SLAN networks use a dynamic, collision free numerical identification tag

for authenticated hosts – *Network ID* (NID). A NID is the digest of two values: the username and the KSK of the (authenticated) user that is using the host.

$$\mathrm{NID}_{host} = \mathrm{digest}\,(\mathit{username}, \mathrm{KSK}_{username,host})$$

The term *username* means some form of user identification used in the 802.1X authentication process. The KDC receives <*username*, *KSK*> pairs from the 802.1X AS, computes a NID from them and keeps a table of <*username*, *KSK*, *NID*> triplets. Authenticated hosts can compute by themselves the NID from their EMSK key (see Figure 2).

Hereafter, we will simply use the expression NID_i for referring the NID of the host h_i and the expression KSK_i for referring the KSK of the user using host h_i.

3.3 KDC key distribution

Session keys distributed by the KDC are associated to pairs of NID values: a NID identifies a session key requester and the other NID identifies the peer that will share that key.

The KDC key distribution paradigm closely follows Kerberos (Kohl and Neuman, 1993) (see Table 1): an authenticated host sends a *Key Request* message for requiring a session key to interact with other (authenticated) host; and the KDC replies with a *Key Reply* message, containing a session key encrypted with the KSK of both peers (messages 1 and 2). The *key propagator* is similar to a Kerberos ticket: it contains the session key and is encrypted with the peer KSK. The requester is responsible for sending the received key propagator to the peer (message 3).

Table 1 Basic protocol for distributing a session key $\mathrm{SK}_{1,2}$, provided by the KDC, from host h_1 to host h_2

Peers	*Main message contents*	*MAC key*
$h_1 \rightarrow$KDC	$\mathrm{NID}_1, \mathrm{NID}_2, \mathit{Options}$	KSK_1
$h_1 \leftarrow$KDC	$\{\mathrm{SK}_{1,2}\}_{\mathrm{KSK}_1}, \mathrm{KeyProp}_{1\rightarrow 2}, \mathrm{NID}_2\ \mathit{info}$	
$h_2 \leftarrow h_1$	*another message*, $\mathrm{KeyProp}_{1\rightarrow 2}$	$\mathrm{SK}_{1,2}$

Notes: $\mathrm{KeyProp}_{1\rightarrow 2} \equiv \{\mathrm{NID}_1, \mathrm{NID}_2, \mathrm{SK}_{1,2}, \mathit{Options}\}_{\mathrm{KSK}_2}$.

KSK_1 and KSK_2 are the secret keys shared between the KDC and the users using hosts h_1 and h_2, respectively; NID_1 and NID_2 represent the NID of hosts h_1 and h_2, respectively.

The key propagator also includes the peer's NIDs and an *Options* field. This field is intended to disseminate, in an authenticated way, relevant characteristics of the host requesting the key propagator. These characteristics may a priori be known by the KDC or may be proposed by the requester. Examples of possible uses are

1 the IP and L2 addresses or

2 relevant administration capabilities (e.g., DHCP server, IP gateway, etc.) of the requester.

The value of the *Options* also depends on the value of the *Options* field in the KDC *Key Request*.

The session key requester may also get, from the KDC, some relevant information about the other host that is going to share the session key. This information is provided in the NID_2 *info* field of the KDC reply, authenticated by the message MAC.

A KDC is free to choose and use a caching policy for the provided session keys. Each time a host requests a new session key for interacting with another host, the KDC generates a new session key or provides a previous one, if still in the cache. This cache is not critical for the system to work, but it can improve its efficiency and resilience to DoS attacks. Naturally, the instantiation of P2P SAs must take this fact into consideration (see Section 4).

Hosts are free to cache session keys, indexed by peer's NID, for the time they may consider more appropriate, and may require new session keys at any time. When a KSK expires, the related NID also expires and cached session keys associated with it become automatically stale.

3.4 Secure DHCP

The authenticated DHCP is presented in Table 2 and runs like this:

1 the client broadcasts a DHCP *Discover* with its NID.

2 a server uses the client NID and its own NID to send a *Key Request* to the KDC.

3 the KDC returns the client username, a session key and a key propagator for the DHCP client.

4 the server uses the username for getting an IP address and sends to the client an authenticated DHCP *Offer* with the key propagator.

5 the client sends an authenticated DHCP *Request* with an extra, authenticated IP-L2 address mapping.

6 the server sends an authenticated DHCP *Acknowledge* with an IP-L2 Propagator – a secure token with the client's IP-L2 address mapping.

The last two messages of the DHCP protocol include, besides the MAC for authentication, two extra values that will be used as a IP-L2 mapping commitment. The DHCP client appends an extra version of the mapping, authenticated with its KSK, to the DHCP *Request*; the DHCP server further authenticates it with its KSK. The final value, returned to the DHCP client, is an *IP-L2 propagator*.

IP-L2 propagators can be fully validated only by the KDC, allowing it to learn current mappings between IP and L2 addresses. This is used for authenticating ARP *Requests* and *Replies*.

Table 2 Secure DHCP

Peers	*Main message contents*	*MAC key*
$h \rightarrow$DS	DHCP *Discover*, NID_h	–
DS$\rightarrow$KDC	NID_{DS}, NID_h	
DS$\leftarrow$KDC	$\{\text{SK}_{DS,h}\}_{KSK_{DS}}$, $\text{KeyProp}_{DS \rightarrow h}$(*Options* = DHCP), NID_h *info* = *username*	KSK_{DS}
$h \leftarrow$DS	DHCP *Offer*, $\text{KeyProp}_{DS \rightarrow h}$ (*Options* = DHCP)	
$h \rightarrow$DS	DHCP *Request*, [IP_h, L2_h] + MAC(KSK_h)	$\text{SK}_{DS,h}$
$h \leftarrow$DS	DHCP *Acknowledge*, IPL2Prop_h	

Notes: $\text{IPL2Prop}_h \equiv [[\text{IP}_h, \text{L2}_h] + \text{MAC}(\text{KSK}_h)] + \text{MAC}(\text{KSK}_{DS})$.

All DHCP messages, except the *Discover*, are authenticated with a MAC computed with a session key ($\text{SK}_{DS,h}$) requested by the DHCP server (DS) and propagated to the client in the DHCP *Offer*. The *Options* field in the key propagator states that it was requested by a DHCP server. The IP-L2 propagator is an authenticated IP-L2 address mapping build together by the DHCP client and server; IP_h and L2_h represent the IP and L2 addresses of the DHCP client, respectively.

3.5 Secure ARP

In the SLAN architecture, ARP caches keep five values, instead of pairs of IP and L2 addresses. The quintets are formed by: the peer NID, IP and L2 addresses, the shared session key and the P2P protection policy.

ARP caches are populated exclusively with

1 information provided by authenticated ARP *Replies* or

2 information gathered for building ARP *Replies*.

Furthermore, during an authenticated ARP run, both hosts should agree on a protection policy concerning their own P2P interaction. This policy agreement is fundamental to set up afterwards a P2P SA between them using the secret key material distributed during the ARP protocol run.

The authenticated ARP is presented in Table 3 and runs like this:

1 h_1 broadcasts an ARP *Request* with its NID, IP-L2 propagator and intended P2P protection policy.

2 The correct responder h_2 uses its NID and the requester's NID to get a new session key from the KDC.

3 The KDC returns a session key, a key propagator and the IP-L2 address mapping of the ARP requester.

4 The responder sends an authenticated ARP *Reply* with the key propagator (providing the P2P session key and assuring the correct IP-L2 mapping of the responder) and the final P2P protection policy.

Table 3 Secure ARP

Peers	*Main message contents*	*MAC key*
$h_1 \rightarrow h_2$	ARP *Request* (*looking for* h_2), NID_1, IPL2Prop_1, *P2P Options*	–
$h_2 \rightarrow \text{KDC}$	NID_2, NID_1, *Options* = $\langle \text{IPL2Prop}_2, \text{IPL2Prop}_1 \rangle$	KSK_2
$h_2 \leftarrow \text{KDC}$	$\{\text{SK}_{2,1}\}_{\text{KSK}_2}$, $\text{KeyProp}_{2\rightarrow 1}(\textit{Options} = \langle \text{IP}_2, \text{L2}_2 \rangle), \text{NID}_1$ *info* = $\langle \text{IP}_1, \text{L2}_1 \rangle$	
$h_1 \leftarrow h_2$	ARP *Reply*, $\text{KeyProp}_{2\rightarrow 1}(\textit{Options} = \langle \text{IP}_2, \text{L2}_2 \rangle)$, *P2P Options*	$\text{SK}_{2,1}$

Notes: The IP-L2 mapping in the ARP *Request* is validated by the KDC. The ARP *Reply* is authenticated with a session key ($\text{SK}_{2,1}$) requested by the responder and included in the key propagator. The *P2P Options* field allows peers to exchange security requirements for setting up a P2P SA, using $\text{SK}_{2,1}$ after the execution of the ARP.

If the KDC keeps a cache of past session keys provided to pairs of hosts, then both hosts may get the same session key on future authenticated ARP transactions, independently of the requester. But if they get a different key, all they have to do is to adjust accordingly the P2P SA mechanisms already being used between them, if any.

4 Setting up LAN-wide SAs

After an ARP protocol run, both interacting hosts on the SLAN share a session key provided by the KDC. This session key can then be used to create a SA to protect the traffic between both hosts. Below we will discuss some possibilities for instantiating SAs at layers 2 and 3 using standard technologies.

The KDC can provide the same, cached session key for the same pair of hosts, or can provide a new, fresh session key on each ARP. Therefore, SLAN hosts must deal with arbitrary refreshments of session keys used in SAs.

4.1 Using link layer SAs

One possibility is to deploy SAs at the link layer for protecting each packet. For instance, one could set up SAs using PPPoE (Mamakos et al.,1999) sessions between pairs of hosts. In this setup, the PPPoE discovery stage can be skipped since each host knows exactly (from the secure ARP) the L2 address of the other host.

The session key distributed along an ARP protocol run can be used to set up the PPPoE session endpoints off-line, namely session identifiers and ECP (encryption control protocol, (Meyer, (1996)) secrets. Upon session key refreshments, the PPPoE sessions are deleted and new ones are created. The P2P *Options* exchanged during the secure ARP must specify ECP negotiation options, such as the possible encryption algorithms.

4.2 Using network layer SAs

Another possibility is to deploy SAs at the network layer, using IPSec (Thayer et al. 1998), for protecting IP datagrams. This is a more natural approach than the previous one because the session key is obtained after an ARP, which means that both hosts will interact using IP (assuming, of course, that the ARP was used to map IP into L2 addresses, which is the usual case in most LANs).

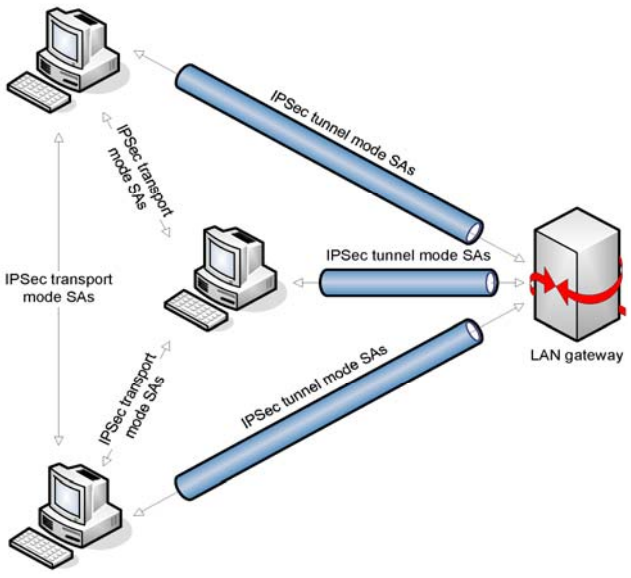
The IPSec SAs can be set up in two different ways:

1 Each host creates an IKE (Harkins and Carrel, 1998) policy using the session key as pre-shared secret. IKE and IPSec SAs between the hosts will then be created automatically between the hosts by the IPSec protocol stack and IKE security policies. Upon session key refreshments, the former IKE policies are deleted and new ones are created. The P2P *Options* exchanged during the secure ARP must specify IKE and IPSec negotiation options.

2 Each host creates two unidirectional IPSec SAs, one for inbound and one for outbound traffic. IPSec SPIs and cryptographic secrets can be derived from the session key. Upon session key refreshments, the former IPSec SAs are deleted and new ones are created. The *P2P Options* exchanged during the secure ARP must specify IPSec operational options supported by IPSec SAs. The prototype described in Section 5 uses this approach.

Using IPSec for LAN-wide SAs has the advantage of using a mature technology for protecting IP traffic, while using the SLAN key distribution protocol to deal with the authenticated set up of IKE or IPSec SAs.

Figure 3 Deployment of IPSec SAs (see online version for colours)



Notes: Between ordinary LAN hosts, transport mode SAs are the simplest ones to use. On the contrary, between those hosts and the LAN gateway, tunnel mode SAs is more adequate.

Using IPSec requires also considering the special case of protecting, only within the LAN, traffic routed through a local gateway between a local host and an external host. In this case, both the local host and the gateway, sharing a session key distributed during ARPs, should deploy an IPSec tunnel to protect routed traffic, instead of using a normal, transport mode IPSec protection (see Figure 3). Nevertheless, each host can easily learn when to use transport or tunnel mode: a gateway always uses tunnel mode; all other hosts use transport mode by default. A host knows when is interacting with a gateway because it receives that information in the *Options* field of a key propagator (c.f. Section 3.3).

4.3 Garbage collection of SAs

Stateless SA endpoints can be deleted when the related ARP cache entries are deleted; they can be recreated again later on, when needed, from the data exchanged in the secure ARP. However, SA endpoints, such as IPSec SAs, tend to be stateful for detecting replays – IPSec SAs include a datagram counter. In this case, the *P2P Options* used along the ARP must specify some initial values for shared state, in order to allow a correct recreation of SA endpoints whenever the KDC provides the same key for the same pair of hosts.

5 Prototype implementation

Just for a proof of concept, we developed a prototype implementing only part of all these components in Linux systems. Namely, we implemented a stand-alone KDC, using long-term <username, KSK> authentication pairs, a secure ARP protocol using user-level processes and data link layer filtering tools and a protection policy for protecting local communications using IPSec. The architecture of the prototype is presented in Figure 4.

The prototype was implemented using DES CBC to encrypt data and SHA-1 to generate MACs. Since the prototype lacks an implementation of the secure DHCP, SLAN servers generate their own IP-L2 Propagators using directly a DHCP KSK.

5.1 Implementation of the KDC

The implemented KDC is just a stand-alone server, i.e., not integrated with any 802.1X authentication server. Thus, in this prototype it uses a configuration file with usernames and passwords to derive KSK and NID values.

The KDC client-server interactions are supported by a protocol implemented on top of ethernet frames. We chose a free ethernet type, 801_H, for our KDC frames. The KDC server acts just like another socket-based server, the difference in this case is that it receives and sends data link layer packets of type 801_H (using a PACKET socket). In this prototype, the KDC provides a new, fresh session key on each request.

5.2 Implementation of the SLAN server

The SLAN server is a user-level process that runs in all SLAN hosts, except the KDC host (see Figure 4). It has three different tasks:

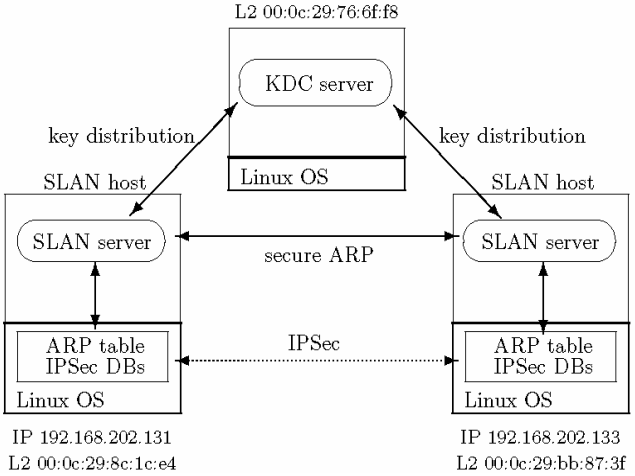
1 Handle all communications with the KDC for getting new session keys.

2 Filter original ARP packets to drop them or to build secure versions from them.

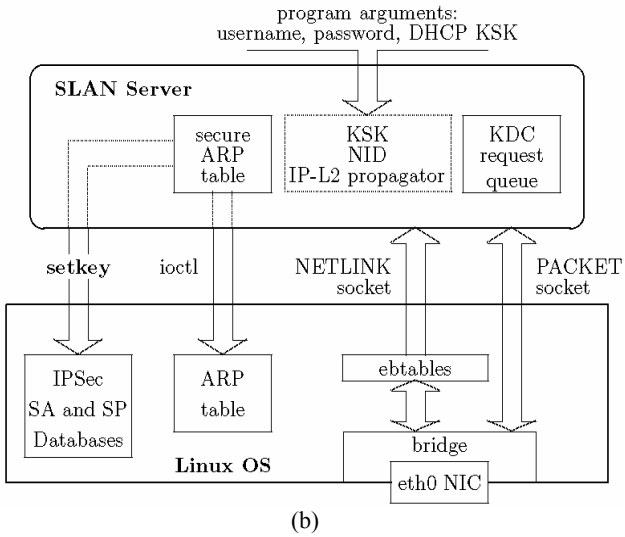
3 Implement a secure ARP protocol, including the setup of kernel ARP tables, using the KDC.

4 Create IPSec SA and security policy (SP) entries from session keys distributed within secure ARPs.

Figure 4 Global and detailed architectures of the prototype implemented, (a) global architecture and (b) detailed diagram of the implementation in each SLAN host



(a)



(b)

The filtering of ARP packets was implemented using ebtables, a tool for managing Linux bridge devices. This tool is used to send to the SLAN server copies of all inbound or outbound ARP packets (both secure and insecure). These copies are provided using the ebtables' ulog watcher and NETLINK multicast sockets.

The filtering action is obtained by dropping the packets after being copied by the watcher. The ebtables rules applied to a bridge interface associated with a network interface eth0 were the following:

```
ebtables -t nat -A PREROUTING -i eth0 -p ARP --ulog -j DROP
ebtables -A OUTPUT -p ARP --ulog -j ACCEPT
```

The first rule sends copies of inbound ARP packets into the SLAN server and drops them. It is used to drop an ordinary ARP *Request* and for processing a secure ARP *Request*/*Replies*; only the KDC will handle inbound ARP packets, and not the Linux OS. The second rule sends a copy of outbound ARP packets into the SLAN server, while letting packets to reach the network. It is used to know when the local kernel is sending an ordinary ARP *Request*, in order to build and send a secure ARP *Request*.

The SLAN server is an event-driven application that waits for data from two sources:

1 a NETLINK socket for getting the packet copies provided by ebtables

2 a PACKET socket for getting KDC replies.

Session key requests executed after receiving a secure ARP *Reply* and before sending a secure ARP *Request* are not synchronous. Instead, all session key requests are registered in a local queue after being sent, and this queue is consulted when a KDC reply arrives.

The SLAN server keeps locally a secure ARP table with the quintets required by the SLAN architecture and manages the Linux OS ARP table accordingly using ioctl calls. In this prototype, we have not yet addressed the management of security policies for protecting traffic between SLAN peers, which should be part of the quintets. Instead, SLAN servers manage one single, hard coded policy, based on IPSec SAs implementing the authentication header (AH) protection mechanism. The reason for using this mechanism is that it allows us to check the enforcement of P2P SAs while being able to observe the original IP datagrams.

The management of IPSec happens when secure ARP entries are updated. When a session key of a secure ARP entry is set, the SLAN server sets up two IPSec SAs for protecting IP interactions with the IP address in the same entry: one for inbound traffic and another for outbound traffic. Both hosts use their IP, the peer IP, the session key and the following algorithm to compute the SA security parameter index (SPI) and cryptographic keys:

$$digest_{1\to 2} = SHA\text{-}1(\mathrm{IP}_1,\mathrm{IP}_2,\mathrm{SK}_{1,2}) \to \begin{cases} digest_{1\to 2}[bits\ 1..32] \to \mathrm{SPI}_{1\to 2} \\ digest_{1\to 2}[bits\ 33..160] \to \mathrm{AH}_{key1\to 2} \end{cases}$$

SPI and key values computed this way are then used to create new IPSec SAs and SP entries. This is done by the SLAN server using the Linux setkey tool. Whenever two hosts get a new session key for refreshing a previous one they had in a secure ARP entry, the existing IPSec SAs between those hosts must be deleted after creating the new ones, with the new session key. The SLAN server also handles this situation, using the old session key and the hosts' IP addresses to compute the SPI of the IPSec SA to delete.

Table 4 shows a traffic capture on the SLAN environment presented in Figure 4(a), executed on top of a VMWare virtual machine. The packet exchange is triggered by a ping from one SLAN host into the other, starting from an empty ARP cache scenario. The 1st packet is the original ARP *Request*, which is captured but not intercepted by the

SLAN server at the sender and is ignored by the receiver. This packet triggers the 2nd packet, the SLAN ARP *Request* (it has more 52 bytes than the former). The 3rd and 4th packets are the KDC *Key Request* and *Key Reply*, respectively. The 5th packet is the SLAN ARP *Reply*, 84 bytes longer than an ordinary ARP *Reply*. The 6th and 7th packets are AH IPSec datagrams containing an ICMP *Echo Request* and *Echo Reply*, respectively. Note that the AH SAs are new, created during the SLAN ARP: the sequence number of AH datagrams is one.

Table 4 Packet capture on a SLAN environment with Linux hosts running on top of a VMWare virtual machine

N.	*Timestamp (m:s)*	*Length (bytes)*	*Source address (L2 or IP)*	*Destination addr. (L2 or IP)*	*Description*
1	39:39.948	42	00:0c:29:8c:1c:e4	ff:ff:ff:ff:ff:ff	ARP: who has 192.168.202.133
2	39:39.949	94	00:0c:29:8c:1c:e4	ff:ff:ff:ff:ff:ff	ARP: who has 192.168.202.133
3	39:40.170	143	00:0c:29:bb:87:3f	00:0c:29:76:6f:f8	ethertype 0x0801
4	39:40.187	143	00:0c:29:76:6f:f8	00:0c:29:bb:87:3f	ethertype 0x0801
5	39.41.517	126	00:0c:29:bb:87:3f	00:0c:29:8c:1c:e4	ARP: 192.168.202.133 is at 0:0c:29:bb:87:3f
6	39:42.239	122	192.168.202.131	192.168.202.133	AH (spi = 0x82fa0685, seq = 0x1) ICMP echo request, id 12095, seq 1
7	39:42.240	122	192.168.202.133	192.168.202.131	AH (spi = 0xe4d0b21d, seq = 0x1) ICMP echo reply, id 12095, seq 1

Notes: The environment is formed by a KDC and two SLAN hosts, all of them with empty ARP caches. Host 192.168.202.131 initiates a ping into host 192.168.202.133, which triggers an ARP transaction. The session key distributed by the ARP is used to create IPSec AH SAs, one for each direction of the communication. These IPSec SAs are then used to protect the ICMP messages.

6 Related work

Some solutions have been proposed to tackle LANs security issues, namely layer two security problems. The scope of the solution for the great majority of proposals resides in protecting only one network protocol, either DHCP or ARP. Examples of solutions are Crypto-Ethernet NIC, secure link layer (SLL), S-ARP, secure ARP and ticket-based ARP (TARP), described below. For an extensive analysis of schemes for detecting and preventing ARP poisoning attacks, see the work of Abad and Bonilla (2007).

Crypto-Ethernet NIC

Khoussainov and Patel (2000) proposed Crypto-Ethernet NIC, combining conventional ethernet network interface card (NIC) functionality with an encryption engine. Network protection is transparent to both user applications and operating system drivers.

In a Crypto-Ethernet NIC environment there is a master key pair kept by a key management unit (KMU). The KMU is used to change and certify public keys of secure NICs in the network. Each Crypto-Ethernet NIC creates a pair of asymmetric keys and knows the public master key. Data is passed through the network in standard ethernet frames, but payloads are authenticated (with the sender's private key) and encrypted (with a session key); the session key goes along with the frame, encrypted with the destination's public key. Only the destination host knowing the proper private key can

decrypt the session key and, hence, the whole packet. The receiver, knowing public key of the sender, can verify the sender's signature to ensure that the message source is genuine. Frames also contain a packet ID field to protect against replay attacks.

When a host wants to know the public key, or the current packet ID, of another host, it sends a request for the destination L2 address, containing its public key certificate, its own packet ID and a nonce; this message is signed with the sender's private key. The destination host sends a reply packet containing its public key certificate, its own packet ID and a nonce; this message is signed with the responder's private key. Receivers can always verify validity of the public key certificates by verifying the master key signature. Nonces help to prevent an active attacker from replaying packets distributed earlier. In addition, to reduce the number of requests for public keys, each NIC has a cache of public keys and last received packet IDs for different source nodes.

The Crypto-Ethernet NIC approach has some management overhead: it requires the existence and management of KMU devices, the manual configuration of public master keys in NICs and the manual certification of NICs' public keys. Furthermore, changes in hardware can be complicated, mainly for other network devices besides hosts – switches, routers, printers, etc. The authors propose a mixed approach, where secure and insecure hosts can communicate to each other through a security gateway.

Secure link layer

Hunleth (2001) proposed SLL for providing authenticated and encrypted communication between any two SLL-enabled hosts on a LAN. SLL requires a certification authority (CA) to produce 'SLL certificates' for all legitimate LAN hosts; these certificates are exchanged between hosts during authentication processes. A host's SLL certificate includes its L2 address, an expiration date and a public key. Each host must have a copy of the CA's public key for validating SLL certificates.

SLL handles authentication and session key exchange before any messages being transferred between hosts. It uses elliptic curve cryptography, namely the MQV (Menezes et al., 1995) secret-sharing protocol, for establishing session keys. In a typical SLL scenario, a host broadcasts an ARP *Request* and the correct responder constructs an ARP *Reply*, but the SLL software queues it and starts a three-step authentication with the requester using MQV. After this mutual authentication, both hosts will share a session key that can be used for encrypting and authenticating ARP *Replies*. SSL performance issues have been pointed out by Bruschi et al. (2003) and Lootah et al. (2005).

S-ARP

Bruschi et al. (2003) proposed S-ARP for protecting against ARP poisoning attacks. It uses asymmetric cryptography and provides message authentication. Any S-ARP enabled host is identified by its IP and has an asymmetric key pair. Certificates provide the binding between hosts' IP address and public key. Certificates are signed by an authoritative key distributor (AKD), a trusted host acting as public key repository.

S-ARP *Requests* are normal ARP *Requests*, but S-ARP *Replies* carry an additional portion at the end – the S-ARP header. This header includes, among other fields, the digital signature of the sender, which is verified by the receiver using the sender's public key. If the receiver does not have the sender's public key, or if the signature cannot be verified with the keys currently held, the public key of the sender is requested from the

AKD. The AKD sends it to the requesting host in a signed message. If the new public key verifies the signature, the reply is accepted and the public key is cached; otherwise, it is rejected. To avoid replay attacks, messages are timestamped and synchronisation messages are exchanged with the AKD.

In networks with IP addresses assigned by DHCP, hosts have null-IP certificates for authenticating S-DHCP transactions and S-DHCP servers propagate IP leases to the AKD, which keeps a mapping between public keys and currently assigned IP addresses.

S-ARP has some management overhead, because manual intervention must occur in the following phases:

1 for setting up the LAN, an administrator must distribute manually the public key and L2 address of the AKD to every host in the LAN

2 when, for the first time, a host connects to the LAN, it generates its private/public key pair and send its signed certificate to the AKD, the correctness of this information must be manually verified by one administrator before it enters AKD repository.

Goyal and Abraham (2005) proposed a modification to S-ARP using digital signatures and one-time passwords based on hash chains to authenticate ARP replies. The S-ARP architecture was kept, but the cryptographic operations are faster.

Secure ARP

Gouda and Huang (2003) proposed secure ARP, another solution for protecting against ARP poisoning attacks. Secure ARP uses a central, secure server that shares secret keys with each local host. This server maintains a database of IP-L2 mappings that is updated periodically through communication with each host. All ARP *Requests* and *Replies* occur between a host and the secure server. The *Replies* are authenticated using the shared keys. Secure ARP is partitioned in two protocols: invite-accept and request-reply.

The invite-accept protocol allows hosts to communicate, periodically and securely, their IP-L2 mapping to the secure server. Invite messages are sent from the secure server to every host in the LAN, whereas accept messages are sent from every host to the secure server. These messages are sent periodically: every T seconds, the secure server sends an invite message to all hosts. Then every host replies by sending back an accept message to the secure server, containing its current IP-L2 mapping. Invite and accept messages contain a nonce, to protect against replay attacks and are authenticated by the sender with the shared key.

The request-reply protocol allows each computer to resolve an IP-L2 mapping. Request messages are sent from any host on the LAN to the secure server, whereas reply messages are sent from the secure server to a specific host on the LAN. When a host needs to resolve an IP-L2 mapping, the host sends a request message to the secure server. This message is authenticated by the shared secret key that binds the host and the secure server. Upon receiving and validating the request message, the secure server searches its database for the corresponding IP-L2 mapping, and then replies with a reply message, containing the requested IP-L2 mapping, authenticated with the same shared key. Request and reply messages also contain a nonce field to protect against replay attacks.

During an initial phase, secure ARP needs a manual configuration of the secret keys shared between hosts and the secure server. The existence of the two new protocols and

the periodic synchronisation between hosts and the secure server also contributes to add some extra complexity to the secure ARP implementation.

Ticket-based ARP

Lootah et al. (2005) proposed TARP, yet another solution for protecting against ARP poisoning attacks. TARP implements security by distributing centrally generated tickets. These tickets transport a host IP-L2 mapping, issued and signed by a local, trusted third party host – the local ticket agent (LTA). Besides the IP-L2 mapping, each ticket also encodes a validity period as an expiration time. Like S-ARP, TARP maintains backward compatibility by integrating the security data in the ARP *Reply*, no changes need take place to the *Request*. The ticket is appended at the end of the ARP *Reply* packet.

In a TARP scenario, IP-L2 address resolution is performed as follows:

1 a host broadcasts an ARP *Request*

2 the host with the requested IP address sends an ARP *Reply*, attaching a previously obtained ticket

3 the requesting host receives the ticket and validates it with the LTA's public key.

If the signature is valid, the address association is accepted; otherwise, it is ignored. The signature on the ticket proves that the LTA issued it, i.e., the IP-L2 address mapping is valid (or at least was when it was issued).

In networks with static IP addresses, TARP hosts need to be manually configured with the LTA public key, an IP address, and a ticket. In dynamic IP networks (DHCP based), TARP authors provide a solution in which the DHCP server acts like a LTA, distributing tickets at the same time as the network parameters. In this case, it is still necessary to manually configure the LTA's public key in each host of the network. So, like S-ARP, also TARP has some management overhead.

With TARP, an attacker can still impersonate a victim by spoofing its L2 address and replaying a captured ticket while being valid. Additionally, because ticket validation is obtained by performing public key cryptography, the cost imbalance between generating a TARP *Reply* and processing it permits DoS attacks by flooding the victim with bogus TARP *Replies*, consuming the victim's CPU resources.

7 Discussion

None of the solutions presented above prevents unauthorised access to the LAN, and consequently the injection of malicious messages. Furthermore, they keep the usual DHCP policies for allocating resources, i.e., based on the L2 address presented by the clients. Also, their goal is to protect specific network transactions, such as ARPs, and not to inter-operate with other protocols. Finally, they require an initial phase where an administrator manually configures keys in each of the LAN hosts.

With the SLAN architecture, we follow a different approach, building up from the 802.1X authentication architecture. Only authenticated and authorised hosts access the network, authenticated hosts can get trustworthy DHCP configurations and IP-L2 mappings, and ARP transactions distribute trustworthy IP-L2 mappings and P2P session keys that can be used by other protocol layers, such as IPSec. The use of a

novel DHCP policy for allocating IP-L2 mappings makes the network more secure against DoS attacks using stolen L2 addresses. Finally, we do not require synchronised clocks.

The S-ARP approach to authenticate DHCP transactions is somewhat similar to the SLAN approach. However, SLAN uses secret, shared keys and MACs, while S-ARP uses asymmetric keys, certificates and digital signatures, which is a slower approach. Furthermore, S-LAN does not address the problem of stolen L2 addresses. Thus, SLAN provides a more efficient and effective approach for dealing with the secure allocation of network resources using DHCP.

The TARP approach, using tickets to distribute IP-L2 mappings, looks similar to the SLAN approach, though being very different. Tickets' signed mappings are valid for a specific time, and can not be revoked before, while SLAN mappings, provided by IP-L2 propagators, are valid while KSKs are in use. Therefore, SLAN provides a better solution for preventing the illegal use of such mappings when no longer valid.

Both Crypto-Ethernet NIC and SLL distribute session keys between LAN hosts, like SLAN. But their strategy is quite different; they use asymmetric cryptography, while SLAN uses symmetric cryptography. Thus, SLAN should perform better, even when considering the deployment of asymmetric algorithms in NICs' hardware.

Finally, secure ARP uses, like SLAN, a central service with secret keys shared with all the hosts in the LAN. However, secure ARP uses a totally new service for this, for managing long-term shared keys, while SLAN extends transparently the 802.1X authentication paradigm for using short-term KSKs (except, possibly, for DHCP servers). Furthermore, secure ARP does not distribute session keys, while SLAN does.

8 Conclusions

In this paper, we presented a holistic solution for achieving cryptographic protection of communications in LAN environments, either wired or wireless. The architecture is build on top of an 802.1X authentication framework. Namely, a KDC is added to an 802.1X AS, allowing KSKs, derived from EMSK keys negotiated with authenticated hosts, to be used for authentication and key distribution in the LAN. Thus, the management overhead for configuring SLAN hosts is minimal if starting from an 802.1X authentication scenario.

Authenticated hosts are identified by NIDs, instead of L2 or IP addresses, and NIDs are computed from usernames and KSKs. Modified DHCP servers, implementing authenticated DHCP interactions, allow a correct configuration of SLAN hosts. An authenticated ARP allows SLAN hosts to get trustworthy IP-L2 mappings and to distribute session keys among LAN peers. Those session keys can then be used by other network protocols to implement other P2P security mechanisms, namely P2P SAs. In this article, we described how PPPoE and IPSec SAs can be deployed from secret material distributed by ARP protocol runs.

LAN-wide, host-to-host SAs are not commonly used, either because they are usually not considered worthwhile or because they are complex to instantiate. But using SLAN, this is achievable with a minor effort, all that is required is some configuration concerning the type of SAs to instantiate and how they should be instantiated.

A prototype implementation was developed for Linux systems. It demonstrated experimentally the suitability of the authenticated ARP and the subsequent IPSec configuration based on the negotiated session keys.

References

Abad, C.L. and Bonilla, R.I. (2007) 'An analysis on the schemes for detecting and preventing ARP cache poisoning attacks', *27th Int. Conf. on Distributed Computing Systems Workshops (ICDCS 2007 Workshops)*, IEEE Computer Society, Toronto, Ontario, Canada.

Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J. and Levkowetz, H. (2004) *Extensible Authentication Protocol (EAP), RFC 3748*, IETF.

Bruschi, D., Ornaghi, A. and Rosti, E. (2003) 'S-ARP: a secure address resolution protocol', *19th Annual Computer Security Applications Conf. (ACSAC '03)*, Las Vegas, NV, USA.

Droms, R. (1997) *Dynamic Host Configuration Protocol, RFC 2131*, IETF.

Dubrawsky, I. (2004) *SAFE Layer 2 Security In-Depth Version 2*, white paper, available at http://www.cisco.com/warp/public/cc/so/cuso/epso/sqfr/sfblu wp.pdf.

Gouda, M.G. and Huang, C. (2003) 'A secure address resolution protocol', *Computer Networks*, Vol. 41, No. 1.

Goyal, V. and Abraham, A. (2005) 'An efficient solution to the ARP cache poisoning problem', *Proc. of the 10th Australasian Conf. on Information Security and Privacy (ACISP '05), Lecture Notes in Computer Science (LNCS 3574)*, pp.40–51, Brisbane, Australia.

Harkins, D. and Carrel, D. (1998) *The Internet Key Exchange (IKE), RFC 2409*, IETF.

Hunleth, F. (2001) *Secure Link Layer*, available at http://www.hunleth.com/fhunleth/projects/sll/sll report.pdf.

IEEE (2001) *IEEE Standards for Local and Metropolitan Area Networks: Port Based Network Access Control*, IEEE Std 802.1X-2001.

Khoussainov, R. and Patel, A. (2000) 'LAN security: problems and solutions for ethernet networks', *Computer Standards & Interfaces*, Vol. 22, pp.191–202.

Kohl, J. and Neuman, C. (1993) The Kerberos Network Authentication Service (V5), RFC 1510, IETF.

Lootah, W., Enck, W. and McDaniel, P. (2005) 'TARP: ticket-based address resolution protocol', *21st Annual Computer Security Applications Conf. (ACSAC '05)*, Tucson, AZ, USA.

Mamakos, L., Lidl, K., Evarts, J., Carrel, D., Simone, D. and Wheeler, R. (1999) *A Method for Transmitting PPP Over Ethernet (PPPoE), RFC 2516*, IETF.

Menezes, A., Qu, M. and Vanstone, S. (1995) 'Some new key agreement protocols providing implicit authentication', *SAC '952th Annual Workshop on Selected Areas in Cryptography*, Ottawa, Canada.

Meyer, G. (1996) *The PPP Encryption Control Protocol (ECP), RFC 1968*, IETF.

Plummer, D. (1982) *Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware, RFC 826*, IETF.

Postel, J. (1981) *Internet Control Message Protocol, RFC 792*, IETF.

Thayer, R., Doraswamy, N. and Glenn, R. (1998) *IP Security Document Roadmap, RFC 2411*, IETF.

Zúquete, A. and Marques, H. (2006) 'A security architecture for protecting LAN interactions', *Information Security Conf. (ISC 2006),* Lecture notes on computer science, Samos, Greece.